Structural Optimization of Turbomachinery Rotors: A Study of Machine Learning Surrogate Models

Kristopher C. Pierson[1] and John L. Gillespie[2]
NASA Glenn Research Center, Brook Park, Ohio 44135

Matthew J. Ha[3]
HX5, LLC, NASA Glenn Research Center, Brook Park, Ohio 44135

Joshua Stuckner[4]
NASA Glenn Research Center, Brook Park, Ohio 44135

The use of machine learning models as surrogates for turbomachinery component design has demonstrated promising results in various previous studies. This includes a previous study by the authors, where Finite Element Analysis (FEA) data was used to train a neural network model, which was then used for optimizing rotor blade design to reduce stress without altering blade thickness. However, many decisions affect the quality and accuracy of the optimization and the effects of these options have not been rigorously tested. This study investigates key factors influencing this process: the selection of surrogate models, the number of design variables, and the training sample size. In this study, the design space is sampled using Latin Hypercube Sampling (LHS). Samples of various sizes ranging from 500 to 5000 are used to train the surrogate models. Two machine learning models from the Python package scikit-learn are evaluated: Random Forest Regressor (RF) and Multi-layer Perceptron Regressor (MLP). The MLP, a neural network, is further analyzed by varying the number of hidden layers from 1 to 5, with each layer containing 100 neurons. Due to the stochastic nature of these models, each is assessed using 100 different random initializations. Model accuracy is first evaluated using a validation set, generated independently from the training set using a separate LHS. This validation set, which is 20% the size of the training data, is also used for early stopping in the neural network models. The accuracy is further evaluated on a separate test set that is never used during any part of the model training process. Finally, the best-performing surrogate model from each type is employed for design optimization using a differential evolution optimization tool from SciPy. The results of the surrogate model optimization are evaluated using FEA to confirm the predicted performance.

[1] Research Aerospace Engineer, Multiscale and Multiphysics Modeling Branch, AIAA Member.
[2] Research Aerospace Engineer, Turbomachinery and Turboelectric Systems Branch, AIAA Member.
[3] Research Design Engineer, Turbomachinery and Turboelectric Systems Branch, AIAA Member.
[4] Research Materials Engineer, Multiscale and Multiphysics Modeling Branch, AIAA Member.

I. Introduction

Surrogate modeling techniques, particularly those involving neural networks, have been shown to have strong potential for optimizing blade geometries for turbomachinery. An early application was demonstrated by Pierret and Braumbussche, who performed aerodynamic design optimization of turbine blades, subject to mechanical constraints, using a neural network surrogate model and a simulated annealing optimization algorithm [1]. Other examples (out of many) include work by Pasquale et al. to perform genetic algorithm-based aerodynamic optimization of blades using both Kriging and neural network models [2], and Persico et al. who used a genetic algorithm with multiple surrogate models to optimize blades for a supersonic Organic Rankine Cycle turbine nozzle [3].

A previous paper by two of the authors [4] presented an optimization methodology which first constructs a neural network surrogate model to approximate Finite Element Analysis (FEA) simulations and then uses the surrogate model for design optimization. Design optimization using this method was then conducted on an existing fan design from the University of Cincinnati called the Tail Cone Thruster Design No. 7-28-8 [5, 6]. The objective function for the optimization minimized peak von-Mises stress predicted by FEA, which resulted in an 80% reduction from the original stress value without changing the thickness of the blade. This promising result has inspired further study on the utilization of neural networks (NNs) and other types of machine learning (ML) models as surrogate models for design optimization.

This study investigates two types of surrogate models for their capability of accurately representing the design space and their optimization performance. The two types of models are implemented using the scikit-learn library [7]; they are:

1. Multi-layer Perceptron Regressor (MLP) [8]
    - Three MLP configurations are evaluated by changing the number of hidden layers (1, 3 or 5) where each layer has 100 neurons.
2. Random Forest Regressor (RF) [9]

Each design space is sampled using SciPy's Latin Hypercube Sampling (LHS) [10] , with the number of samples ranging from 500 to 5000. The accuracy of ML models trained with the various number of samples is then evaluated with the expectation that a larger number of samples will increase the accuracy of the ML model. The number of samples required to accurately represent the design space is expected to be influenced by the number of design variables. Therefore, three design spaces are studied, with 4, 6, and 8 design variables, respectively. Since all models considered in this work are stochastic in nature, 100 models are trained for each configuration using different random seeds to initialize the training. For each training set, independent validation sets that are 20% the size of their associated training set are generated using LHS. The validation sets are used for early stopping in the MLP training. The best-performing model from each group, as determined by its validation set performance, is selected for evaluation of its optimization performance.

For each of the 3 design spaces studied, a test data set with 2000 sample points is created to assess the models that are selected for further evaluation. This test set is not used during any part of the training of the models. Finally, the models developed in the prior sections are put to the test as surrogate models for design optimization. All optimization results are compared against FEA simulations to validate the response predicted by the ML models at the optimal point.

II. Description of the Study

The process of training the surrogate models as outlined in a flowchart in Fig. 1. This paper studies steps 1, 2 and 4. Step 1 is studied by varying the number of design variables (DVs) in the design space. Step 2 is studied by varying the number of sample points in the LHS. Step 4 is studied by evaluating different types of machine learning models for their ability to accurately represent the design space. The details of the study are described in the following sub-sections.

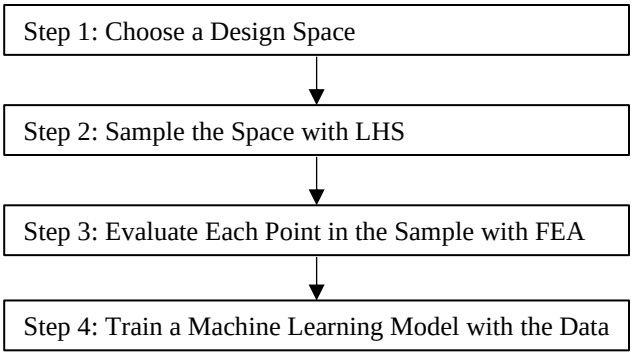

Fig. 1: Flowchart showing the process of training the machine learning models studied in this paper.

A. Baseline Design and The Design Spaces

Designs are generated using T-Blade3 [11]. The baseline design for this work is the Tail Cone Thruster Design No. 7-28-8 [5, 6], which is openly available on the T-Blade3 GitHub page. The baseline design for this incorporates a minor modification from the T-Blade3 GitHub design: the lean and sweep control points at 0% span are set as zero. Additionally, these lean and sweep control points at 0% span are excluded from consideration as design variables in all evaluated design spaces. This model uses a large wedge geometry as a stand-in for a disk as the focus of this work is on the blade design rather than disk design. All geometries have a $1/16$ inch fillet applied at the root of the blade. This fillet size was chosen somewhat arbitrarily, selected primarily because it is commonly available in manufacturing and appears proportionally appropriate relative to the blade's chord and span. The fillet is applied using Ansys DesignModeler.

The design variables to be studied are lean and sweep control points at various locations along the span of the blade. These parameters affect how the blade sections are stacked. Sweep control points are defined in the non-dimensional m_s' coordinate (meridional coordinate), which is formulated using the expected streamlines of the flow. The design parameters for sweep represent a deviation from an existing geometry and are referred to as $\Delta m_{sp}'$, where sp defines the spanwise location of the control point (i.e.. $\Delta m_{25}'$ means that the control point is at 25% of the rotor span). Lean control points produce a shift of the blade profile in the θ coordinate, which is rotation about the x-axis and has units of radians. Similarly, the lean control points also represent a deviation from existing geometry and are referred to as $\Delta\theta_{sp}$ where sp again defines the spanwise location of the point. These control points are distributed along the span to shape the blade profile, and they form a smooth, continuous geometry when interpolated using a B-Spline curve. This B-Spline formulation leverages the control points to gradually transition the geometry, avoiding abrupt changes and ensuring smoothness across the blade surface. Additional explanation of these design variables and the coordinate system within which they operate is available in the authors' previous work in Section VI of Ref. [4]. A depiction for explanatory purposes is given by Fig. 2 below, which is Fig. 5 in Ref. [4].

Fig. 2 shows the effect of altering a lean or sweep control point at the 25% span. In Fig. 2a, $\Delta m_{25}'$ is changed by a value of 0.2 to show an exaggerated effect of changing this parameter. Fig. 2b shows a similar example of changing $\Delta\theta_{25}$ by a value of -0.2.

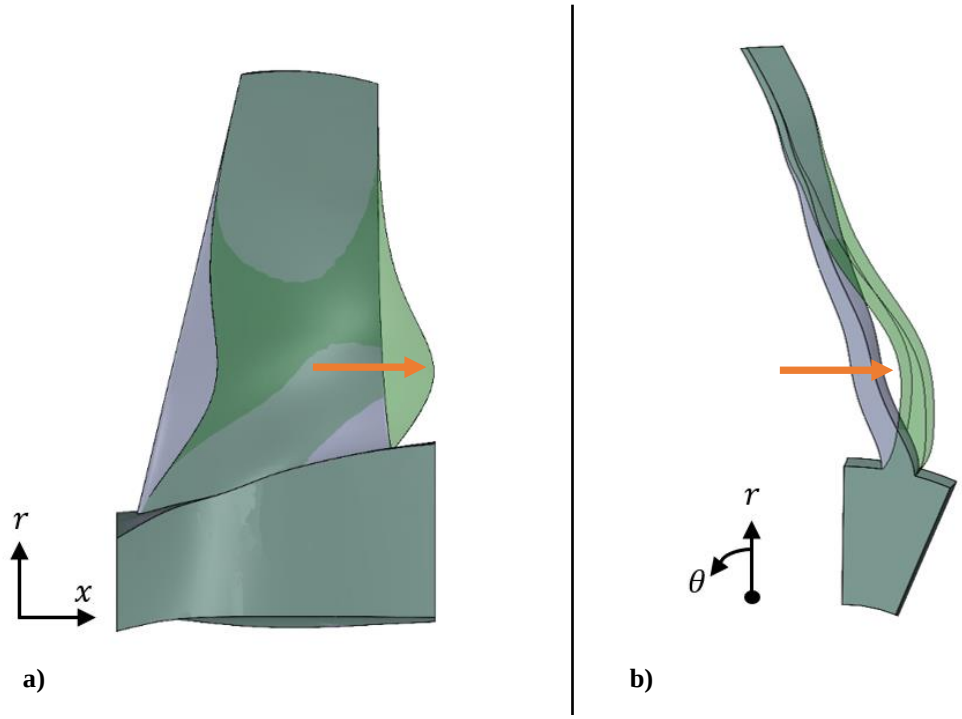

Fig. 2 Sweep (a) and lean (b) examples; a) positive sweep control point at 25% span ($\Delta m'_{25} = 0.2$); b) negative lean control point at 25% span ($\Delta\theta_{25} = -0.2$)

Three design spaces are studied in this paper. The three spaces are very similar to one-another, with the difference being the number of spanwise lean and sweep control points used to modify the geometry from the baseline. Each span location has both a lean and sweep control point. In Design Space 1, only two span locations are assigned control points, while Design Spaces 2 and 3 use progressively more, with three and four spanwise locations respectively. The increase in spanwise control points introduces finer granularity in controlling the blade's lean and sweep adjustments while also increasing the complexity of the design space by pushing it to higher dimensions. There are expected advantages and disadvantages to increasing the number of control points. Although additional control points may enable superior design outcomes, they also complicate the optimization process, as the surrogate model must account for a larger design space with increased interactions among design variables. Consequently, accurately representing these higher-dimensional spaces is expected to require a larger number of samples. Every variable in each design space has the same range of (-0.17,0.03). This range is chosen to encompass both the baseline design and the expected optimal value for each design variable. The expectation for the optimal values is based upon previous work in Ref. [4]. The three design spaces are shown below in Table 1.

Table 1: The three design spaces that are studied in this paper.

Design Space 1	
Variable	Range
$\Delta m'_{50}$	(-0.17,0.03)
$\Delta m'_{100}$	(-0.17,0.03)
$\Delta\theta_{50}$	(-0.17,0.03)
$\Delta\theta_{100}$	(-0.17,0.03)

Design Space 2	
Variable	Range
$\Delta m'_{25}$	(-0.17,0.03)
$\Delta m'_{50}$	(-0.17,0.03)
$\Delta m'_{100}$	(-0.17,0.03)
$\Delta\theta_{25}$	(-0.17,0.03)
$\Delta\theta_{50}$	(-0.17,0.03)
$\Delta\theta_{100}$	(-0.17,0.03)

Design Space 3	
Variable	Range
$\Delta m'_{25}$	(-0.17,0.03)
$\Delta m'_{50}$	(-0.17,0.03)
$\Delta m'_{75}$	(-0.17,0.03)
$\Delta m'_{100}$	(-0.17,0.03)
$\Delta\theta_{25}$	(-0.17,0.03)
$\Delta\theta_{50}$	(-0.17,0.03)
$\Delta\theta_{75}$	(-0.17,0.03)
$\Delta\theta_{100}$	(-0.17,0.03)

Note that the number of control points can be easily altered in the T-Blade3 input file. So, when investigating two spanwise locations of control points (four DVs total), the setting for the total number of control points in the T-Blade3

input file is set as 3 for both lean and sweep control points. Two of these locations have design variables (control points at 50% and 100% span), while the 0% span location is not a design variable and is always held at zero.

B. Data Sampling and Geometry Generation

Latin Hypercube Sampling (LHS) is used to sample the design space. This is done using SciPy's LHS implementation [10]. The LHS method is described in Ref. [12] as "Latin hypercube designs have one-dimensional uniformity in that, for each input variable, if its range is divided into the same number of equally spaced intervals as the number of observations, there is exactly one observation in each interval." However, it is still possible that point bunching may occur as each combination of variables is randomly generated.

To improve the sample, the 'random-cd' optimization method included with the SciPy LHS implementation is utilized. The 'random-cd' methods seeks to lower the centered discrepancy [13]. Centered discrepancy is a calculation of the largest difference between the number of sample points in any sub-interval of the space and the expected number of sample points within that location. For example, if a given sub-interval makes up 20% of the space, it should have approximately 20% of the points. The 'random-cd' method uses random permutations of the sample to reduce the centered discrepancy, leading to a more uniform sample across the design space. The SciPy implementation of both MLP and RF also has the option to introduce a seed number so that the sample generated can be reproduced. Each sample used in this study has a different seed number.

To generate training data, each design space is sampled with separate samples of size 500, 1000, 2000 and 5000. Every design space sample also has a validation fraction associated with it, which is an independent LHS that is 20% the size of the training set. Finally, each design space has one LHS test set that contains 2000 sample points. The test set is not used in any part of the training while the validation set is used for early stopping of the MLP models.

After the LHS of the space is created, the geometry is generated by a Python program that controls T-Blade3, which outputs IGES files of the blade geometry for each sample design. The next step after generating the geometry is the evaluation of its response to a centrifugal load using FEA.

C. FEA Methods and Conditions used to Generate Training Data

The FEA is conducted in an automated fashion using a Python program that controls Ansys Workbench. The program follows the following process to determine stresses for a new geometry.

1. Start Ansys Workbench.
2. Load a template case.
3. Change the geometry to that of the sample design.
4. Mesh and simulate the new geometry.
    - Based on parameters in the template case.
5. Store the result in a CSV file.

Each case is meshed with the same parameters. A mapped face meshing technique is applied to the blade face and fillet, with 70 divisions along both the chord and span and 5 divisions across the fillet radius. Occasionally, at certain sample points, the mapped meshing fails to generate correctly. When this happens, the Ansys meshing tool automatically skips face mapping for that instance. Other mesh parameters are carefully configured to ensure consistent refinement across all cases, whether mapping is successful or not. The results show minimal impact from the lack of mapping, with the main advantage being that successful face mapping significantly speeds up mesh generation. Global mesh sizing parameters in Ansys Mechanical are set as show in Table 2. No mesh refinement study is provided in the current work, but mesh parameters are similar to the mesh used in Ref. [4], which included a mesh study on this same geometry.

Table 2: Ansys global mesh sizing parameters

Element Size	0.0025 [m]
Use Adaptive Sizing	No
Growth Rate	1.85
Max Size	0.001 [m]
Mesh Defeaturing	No
Capture Curvature	No
Capture Proximity	Yes
Proximity Min Size	0.001 [m]
Proximity Gap Factor	3
Proximity Size Source	Faces and Edges

The mesh settings as described in the preceding paragraph and Table 2 result in approximately 160,000 nodes and 100,000 elements in the mesh. A representative mesh where mapping works on all faces is shown in Fig. 3 with both a view of the full span and a zoomed image near the fillet. Fig. 4 shows a similar set of images where the face mapping is deactivated on one of the blade faces. The mesh in Fig. 4 is a similar number of nodes and elements as the one in Fig. 3. The FEA prediction of max von-Mises stress varied between the two types of meshes by less than 2%.

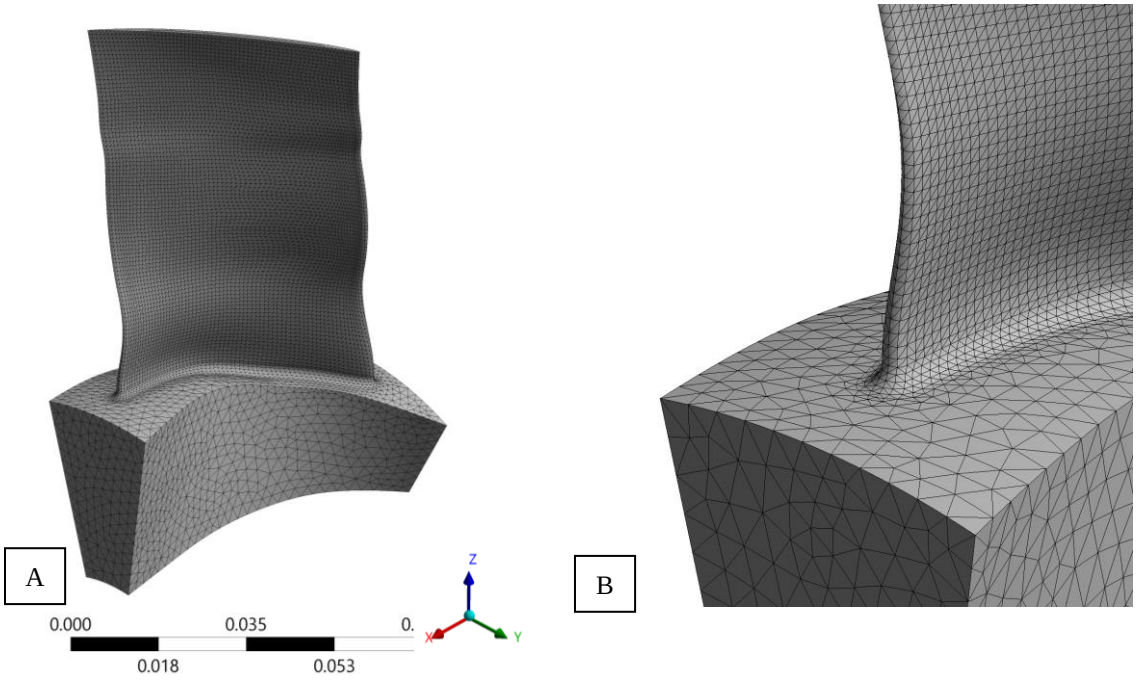

Fig. 3: Images of the FEA mesh where mapped face meshing worked on all faces. A) View that shows the length of the whole blade. B) Zoomed view of the fillet region.

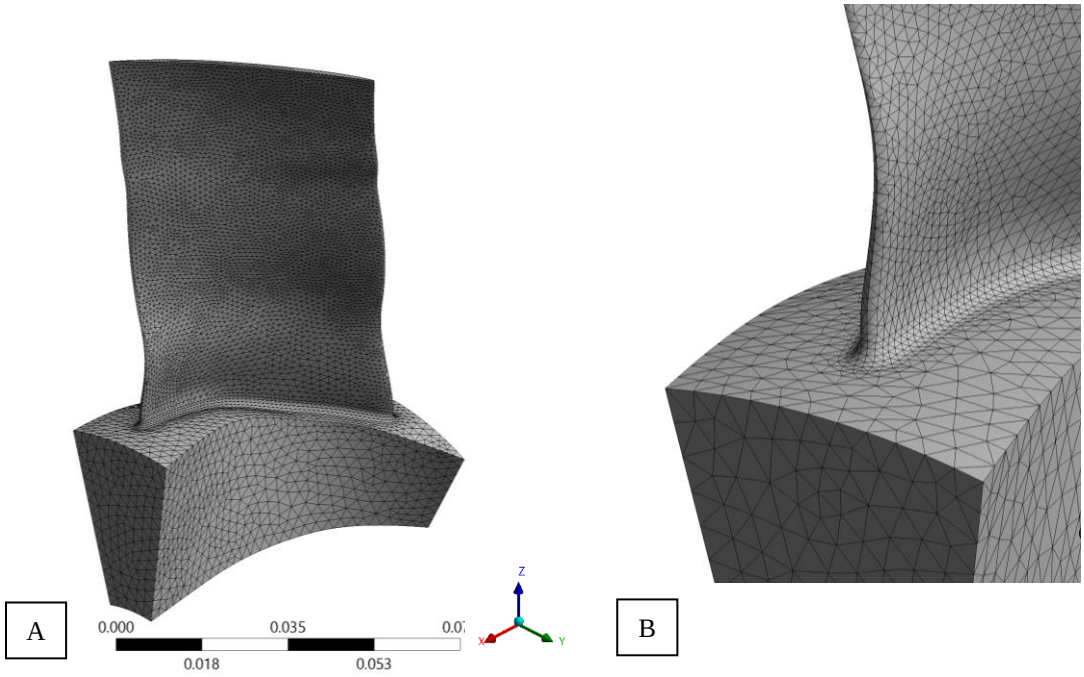

Fig. 4: Mesh example where face meshing is inactive on the blade face. A) View that shows the length of the whole blade. B) Zoomed view of the fillet region.

The material applied to the FEA model is titanium Ti-6Al-4V. The properties for this material are shown in Table 3.

Table 3: Material Properties of Ti-6Al-4V

Density [kg/m^3]	4430
Young's Modulus [Pa]	1.14E+11
Poisson's Ratio	0.33
Tensile Yield Strength [Pa]	1.10E+09

The simulation conditions for the model are as follows: the base of the wedge is fully fixed, and the rotational velocity is 21581 RPM. The automated simulation method outputs the result in terms of margin of safety (MS) based on the tensile yield strength of the material (see Section II.E below for the definition of MS). The MS and the design parameters for each sample are then used as the training data for the ML models. Due to license and computing limitations, each sample is simulated sequentially. Therefore, the cost of generating the training data scales linearly with sample size. Approximately 1000 simulations can be conducted in a single day.

D. Machine Learning Models

Two types of machine learning models from scikit-learn are tested as surrogate models to represent the design space response surface [7]. The first model is a multilayer perceptron (MLP) neural network, generated using the MLPRegressor class [8]. Each MLP model studied contains layers with 100 neurons per layer, tested across three configurations: a single layer, three layers, and five layers. A study on varying the number neurons per layer was not included in this work to keep the scope of the study reasonable. Layer size could be an interesting topic for future research.

The second model studied is a random forest (RF) model generated using the RandomForestRegressor class [9]. Two configurations are tested for the RF model: one with 100 decision trees and another with 1000 decision trees.

The functions for training both model types are included in the Appendix. Training the RF model is straightforward, relying on the fit method provided by the class. For the MLP models, a custom training loop with early stopping was implemented to prevent overfitting. Early stopping halts training if no improvement on the

validation set is observed after 100 epochs, ensuring that the best-performing model on the validation set is selected and saved. The MLP training function is also configured to allow a maximum of 20,000 epochs, though no model required that many.

For each combination of model type and training data, 100 models are trained to assess the impact of the stochastic nature of the training process. Both scikit-learn classes studied provide a parameter to control the random seed, which was varied from 1 to 100.

E. The Objective Function

The objective function used for design optimization is the margin of safety (MS) based on the yield strength, which is defined below in Eq. (1). The optimization seeks to maximize MS.

$$\text{MS} = \frac{\text{Yield Strength}}{1.1 \cdot \text{Max Stress}} - 1 \tag{1}$$

III. Baseline Design FEA Result

The baseline design, described in Section IIA, using the mesh settings and simulation conditions in Section IIC resulted in a MS of -0.0545. The negative value for margin of safety indicates that the initial design has stress that is slightly higher than is permitted for this design. Filled contours of von-Mises stress on the blade are shown below in Fig. 5. The visualization shows that the maximum stress occurs in the filleted region.

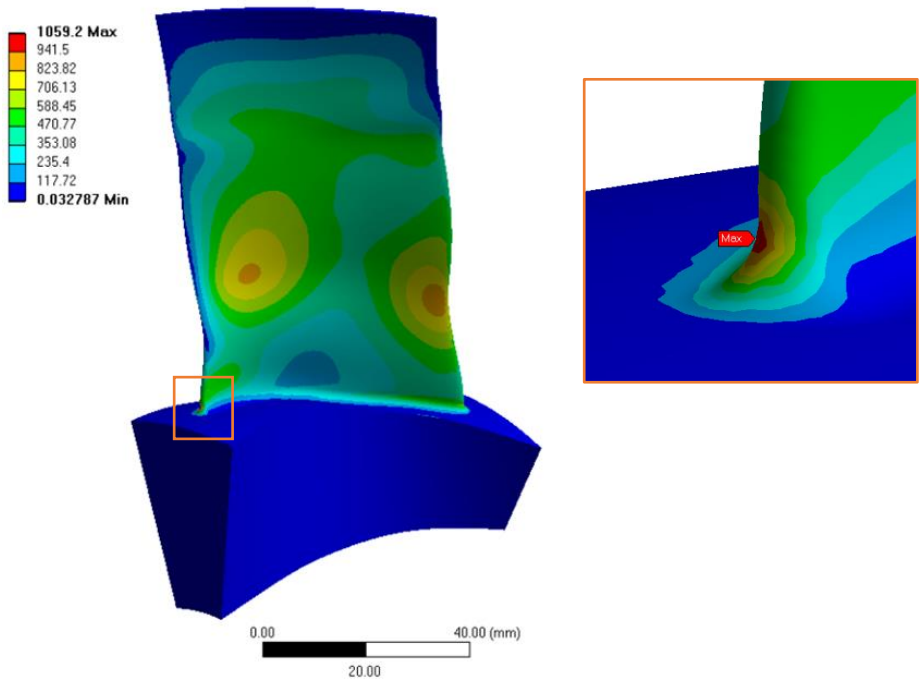

Fig. 5: Blade von-Mises stress for the baseline model. The region of max stress is zoomed. The zoomed region is indicated in an orange box in the full model view.

IV. Model Evaluation Methodology

Models are initially evaluated using validation data, which would typically serve as the basis for selecting the best model if only training and validation data were available. This approach aligns with how the method would be applied in a real-world scenario since producing a test set is computationally expensive. A representative result is shown in Fig. 6, which is a three-layer MLP trained using the 5000 sample training set.

To evaluate the performance of the model using the validation data, the coefficient of determination (R^2) is used [14]. For a machine learning model aimed at predicting FEA output, R^2 serves as a measure of how well the predictions match the actual FEA values. An R^2 score of 1 means a perfect fit, with all predictions falling precisely on the line $y = x$, indicating the model captures all variability in the data. A score of 0 shows that the model's predictions are no better than using the mean of the FEA outputs, and a negative R^2 implies that the predictions trend the opposite of the true result. This makes R^2 a useful metric for evaluating predictive power, with higher values confirming stronger agreement between predictions and real values, critical for validating model accuracy. In the following sections, the R^2 value is computed for all trained models on both the validation and test sets.

The best performing model was chosen by the R^2 value of the validation set. As expected, the best performing models for both MLP and RF are the ones trained with 5000 samples. The best performing MLP model has 3 layers, and the best performing RF model has 100 decision trees. The evaluation of these two models using the R^2 metric is shown in Table 4. Based on this evaluation, the performance of the MLP model far exceeds the RF model.

Table 4: Best MLP model and best RF model performance on training, test and validation sets.

	Best MLP	Best RF
Training Set R^2	0.9993	0.9751
Validation Set R^2	0.9944	0.8175
Test Set R^2	0.9919	0.8076

A visualization of the prediction compared to the true FEA result is shown in Fig. 6. In the figure, it is shown that the RF model does not generalize nearly as well as the MLP model. Consistent with the R^2 values in Table 4, the training data for the RF model is predicted much better than either the test data or the validation data (see Fig. 6C and D). This is especially true in regions of high MS, which is unfortunate since the usage of the model in design optimization seeks to maximize MS.

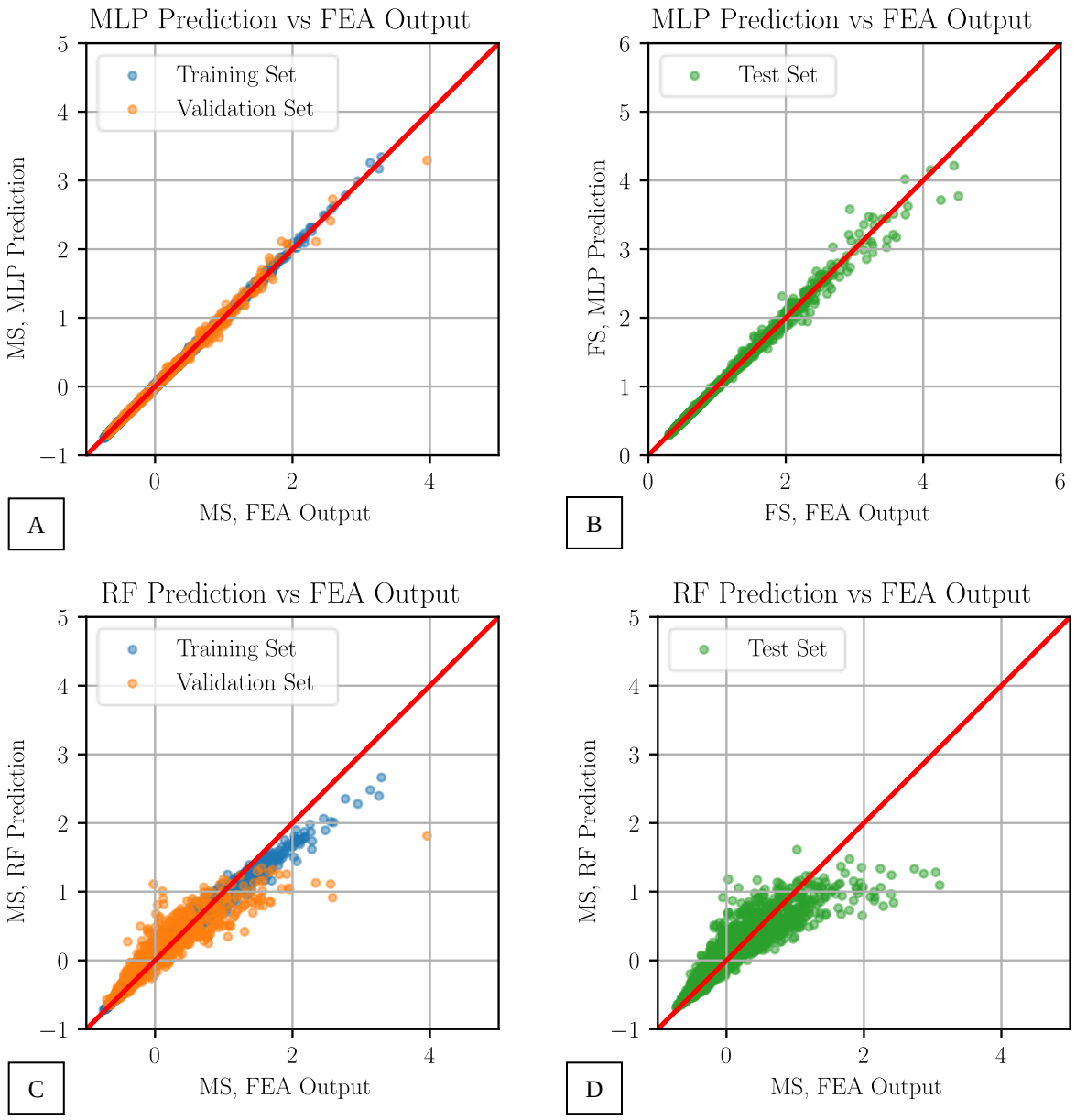

Fig. 6: Model prediction vs FEA output for the best 5000 sample MLP and RF models based on the validation set. A) MLP model evaluated using training and validation set; B) MLP model evaluated using test set; C) RF model evaluated using training and validation set; D) RF model evaluated using test set.

V. Model Evaluation for 6 Design Variable

In this section the accuracy of the two types of models is assessed using two metrics. First by evaluating the performance of the models on the validation and test sets. Second by evaluating the models by plotting a slice of FEA results and comparing that to the slices predicted by the MLP and RF models.

A. MLP Models, R^2 Evaluation

Fig. 7 presents the evaluation of all MLP models using R^2 for both validation and test sets. In Fig. 7, the red line represents the ideal relationship between the test and validation sets, in contrast to its purpose in Fig. 6. Here, the red line indicates that, ideally, performance on the validation set would directly correspond to performance on the test set. Such a relationship implies that improvements in validation set accuracy translate to corresponding improvements on

the test set, suggesting a more accurate representation of the response surface. The results generally show that higher R^2 values on the validation set correspond to higher R^2 values on the test set, despite some variance. Therefore, designers can make confident model selections using only validation data since a test set is not typically available in practice. In this study, the test set evaluates the various models consistently using data not included in any part of model training. An additional benefit of using a dedicated test set in an academic context is that it remains constant across models trained with different sample sizes, providing a consistent basis for comparison despite variations in training and validation set sizes. In practice, this approach is not employed due to the high computational cost associated with generating a dedicated test set; resources are typically better utilized by expanding the training set. It should also be noted that validation accuracy is likely to overestimate real world accuracy when it is used for model selection.

Comparing Fig. 7A with Fig. 7B, it is shown that the model performance for every sample size increases significantly when increasing the number of layers from one to three. The three layer models with the 500 sample training set generally perform as well on the test set as the one layer models created with the 1000 sample training set. The increase from three layers to five layers does not show a significant difference in test set performance.

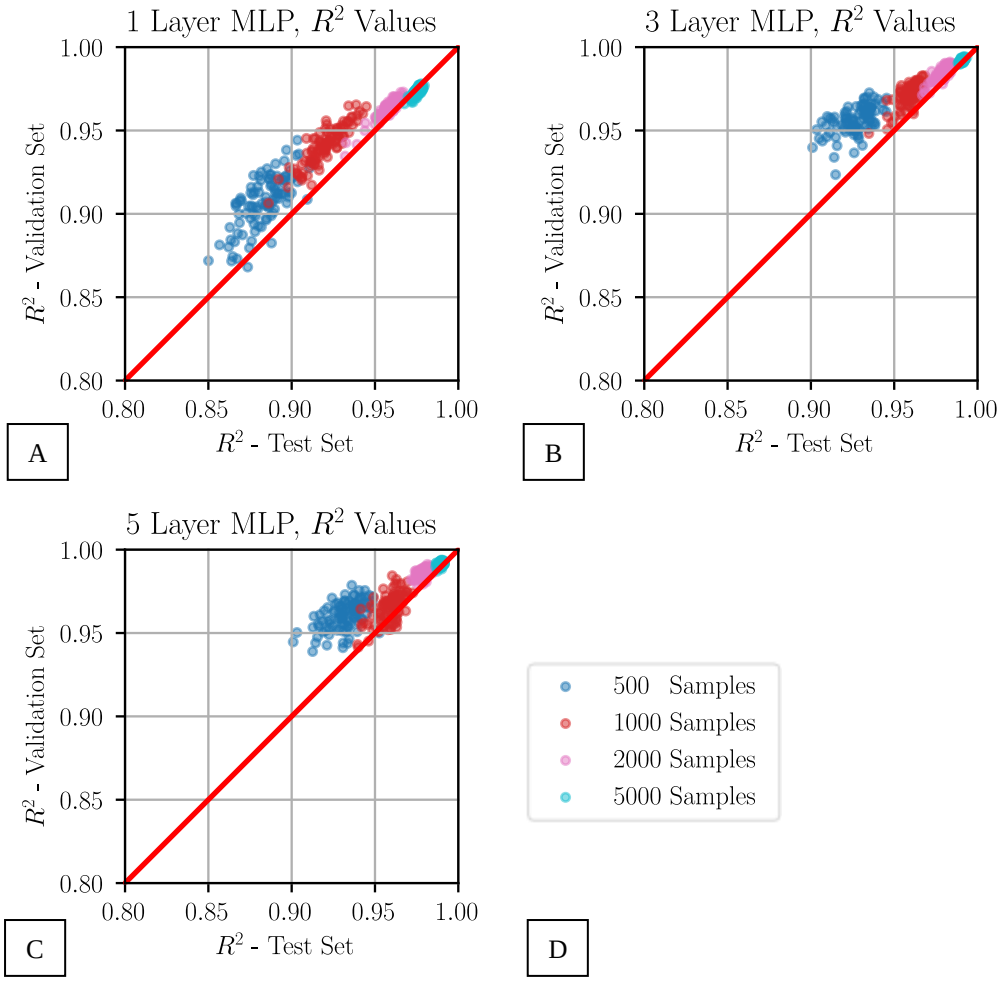

Fig. 7: R^2 values of validation vs test set for all MLP models with 6 DVs. A) 1 layer models; B) 3 layer models; C) 5 layer models; D) Legend showing coloring based on the number of training samples.

Fig. 8 shows a direct comparison of models with the three different layer configurations. Fig. 8A shows models trained with the 500 sample training set. This result shows a very clear benefit of increasing the number of layers from one to three. It also shows that the 5 layer models slightly outperforms the 3 layer models. Fig. 8B shows models trained with the 5000 sample training set. Again, a very clear benefit is seen by increasing the number of layers from one to three. However, this 5000 sample result shows a contradiction with the 500 sample result in terms of the benefit

of three vs. five layers. In the case of the 5000 sample set, the three layer model slightly outperforms the five layer model on the test set. Though, it should be noted that the difference between three and five layers is much less than either of those models compared to the one layer model. Based on this evaluation of the result, the best MLP model for the 500, 1000 and 2000 sample training set is a five layer model while a three layer model produced the best result when trained with the 5000 sample training set. Training time was not timed in this study, but it was negligible compared to training data generation for all models.

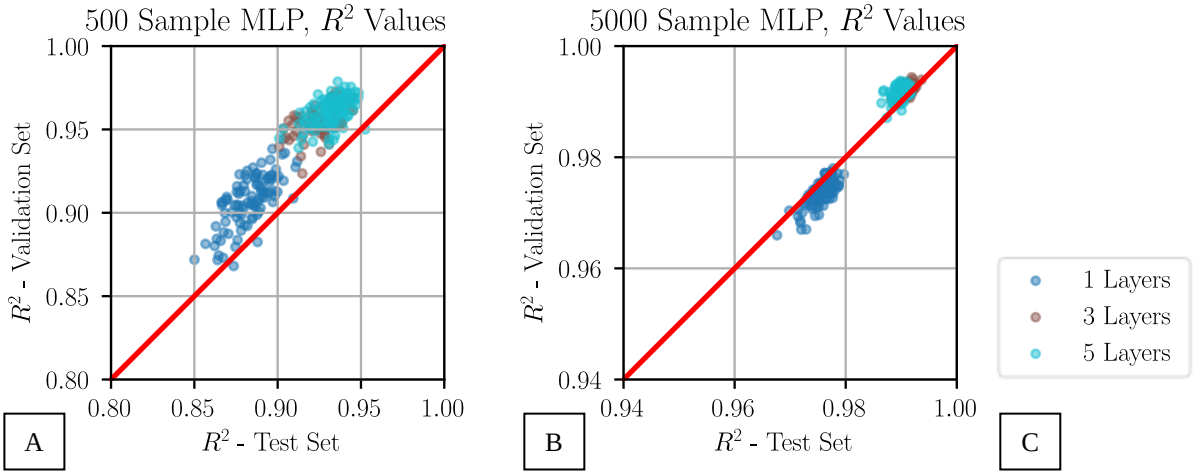

Fig. 8: Comparison of the three MLP model types, each trained using 6 design variables, based on R^2 values of the validation and test sets. A) MLP models trained with the 500-sample set; B) MLP models trained with the 5000-sample training set. C) Legend showing coloring based on the number of layers in the model. Please note that the range of the axes is different between A and B.

B. RF Models, R^2 Evaluation

Fig. 9 shows the accuracy assessment of the RF models with 6 DVs, again using the validation set and the test set R^2 values as evaluation metrics. Please note that there is a large scale difference between Fig. 8 and Fig. 9. The performance of the RF models is surprisingly poor; the best RF model trained with 5000 samples attains a lower R^2 value ($R^2 = 0.82$) than the poorest MLP model trained with only 500 samples ($R^2 = 0.86$). As seen in Fig. 6C and D, the RF models struggle with predicting designs with higher MS.

Fig. 9 also shows that for the RF models, there is no clear relationship between validation set performance and test set performance based on R^2 value. This was initially surprising to the authors since the validation set is not used in any portion of the model training for the RF models, while it is used as early stopping criteria for the MLP models. Consequently, one might anticipate an even stronger relationship between RF performance on the validation and test sets. However, this result is more likely driven by the smaller size of the validation sets compared to the test set. This causes a greater variance in validation set R^2 as the smaller samples make the random variation between models more apparent. With the larger test set, this variance is reduced. It can also be observed that the greater the validation set size, the less variance in R^2 value.

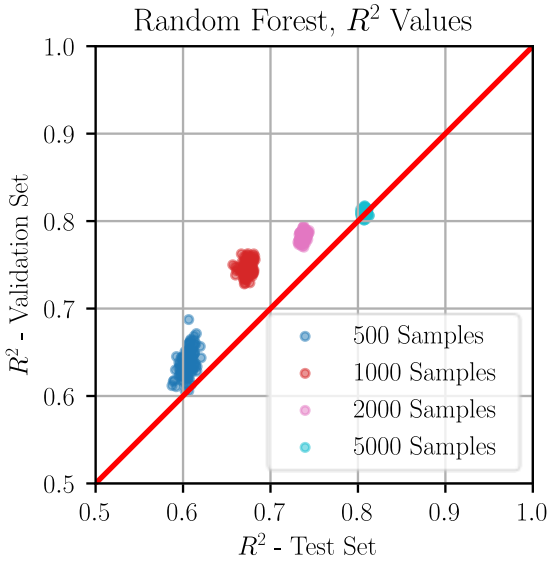

Fig. 9: R^2 values of validation vs test set for all RF models with 6 DVs.

C. Design Space Representation

Another method for qualitatively evaluating ML model performance is to directly compare the shape of the response surface generated by the FEA results with the shape predicted by the ML models. For this comparison, 121 FEA simulations were conducted to create a 2D slice of the response surface. When generating this slice, all other variables were fixed at -0.07, while $\Delta m'_{100}$ and $\Delta\theta_{100}$ were varied across the design space range. Fig. 10 shows the slice of the response surface generated by the FEA output.

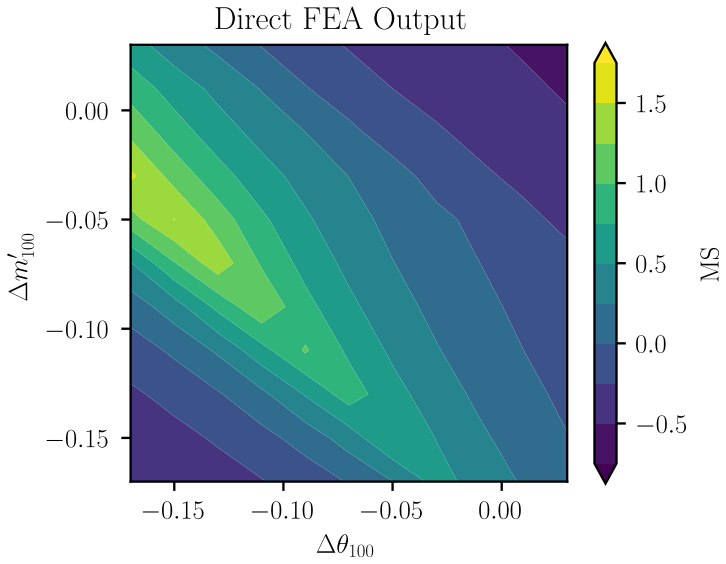

Fig. 10: Slice of the design space showing the response to $\Delta m'_{100}$ and $\Delta\theta_{100}$ while all other variables were help at a constant value of -0.07.

Fig. 11 shows the same slice as predicted by the MLP models trained with different numbers of samples. Each of the slices appears qualitatively similar to the direct output from the FEA, making it difficult to draw conclusions about the accuracy.

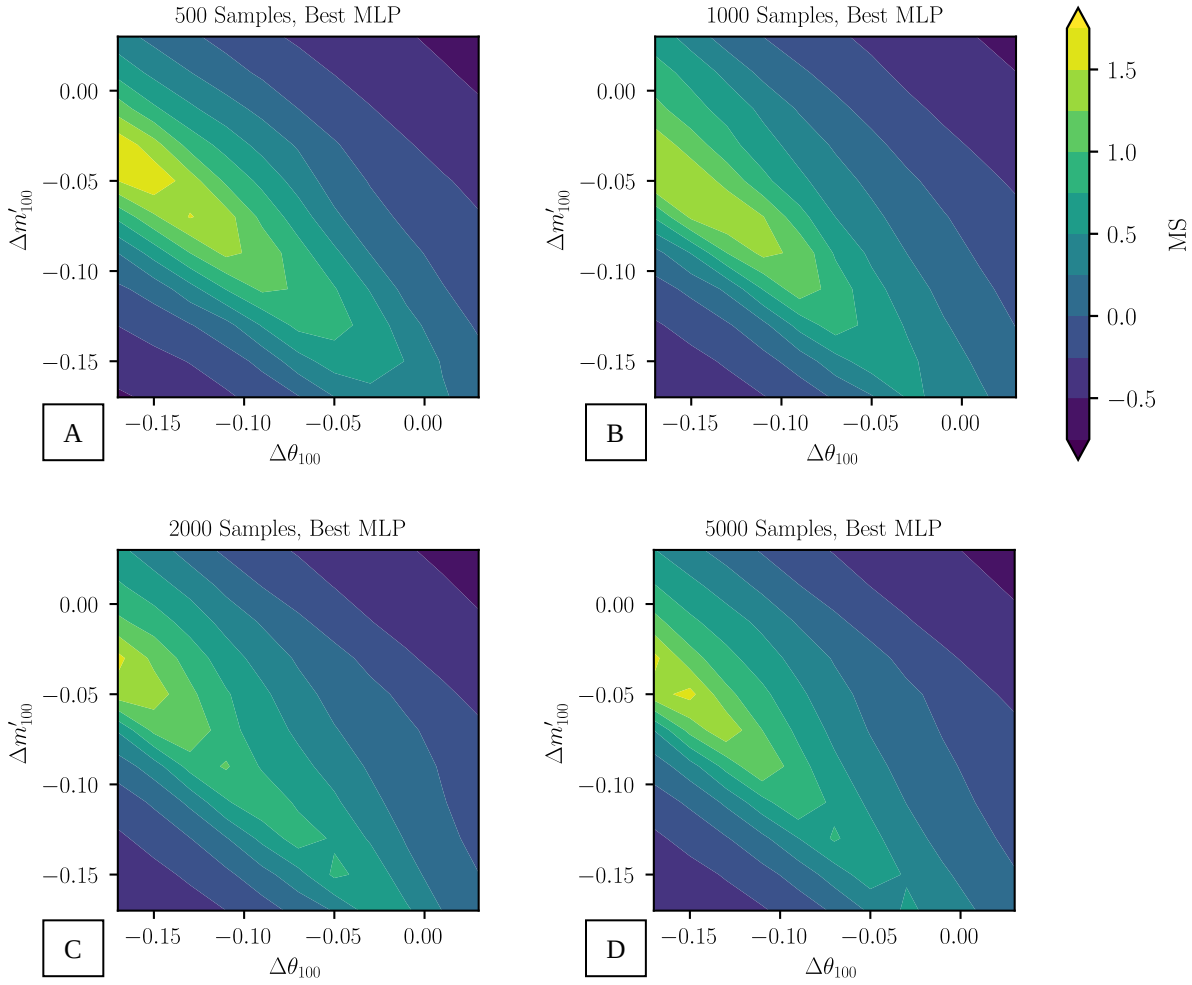

Fig. 11: Design space slices predicted using the best MLP models for each training set. A) 500 sample training set; B) 1000 sample training set; C) 2000 sample training set; D) 5000 sample training set.

Since the MLP models appear so close to the FEA output, a second set of plots, shown in Fig. 12, was created showing the difference between the MLP prediction and the FEA output. In Fig. 12, a positive value of MS difference indicates that the MLP model is over-predicting the MS in the MLP model. The result clearly shows the accuracy of the MLP model increasing with the increased number of samples used for training, which is the expected result. For the model trained with 5000 samples shown in Fig. 12D, the deviation in MS from the FEA is less than 0.1 for the majority of the slice.

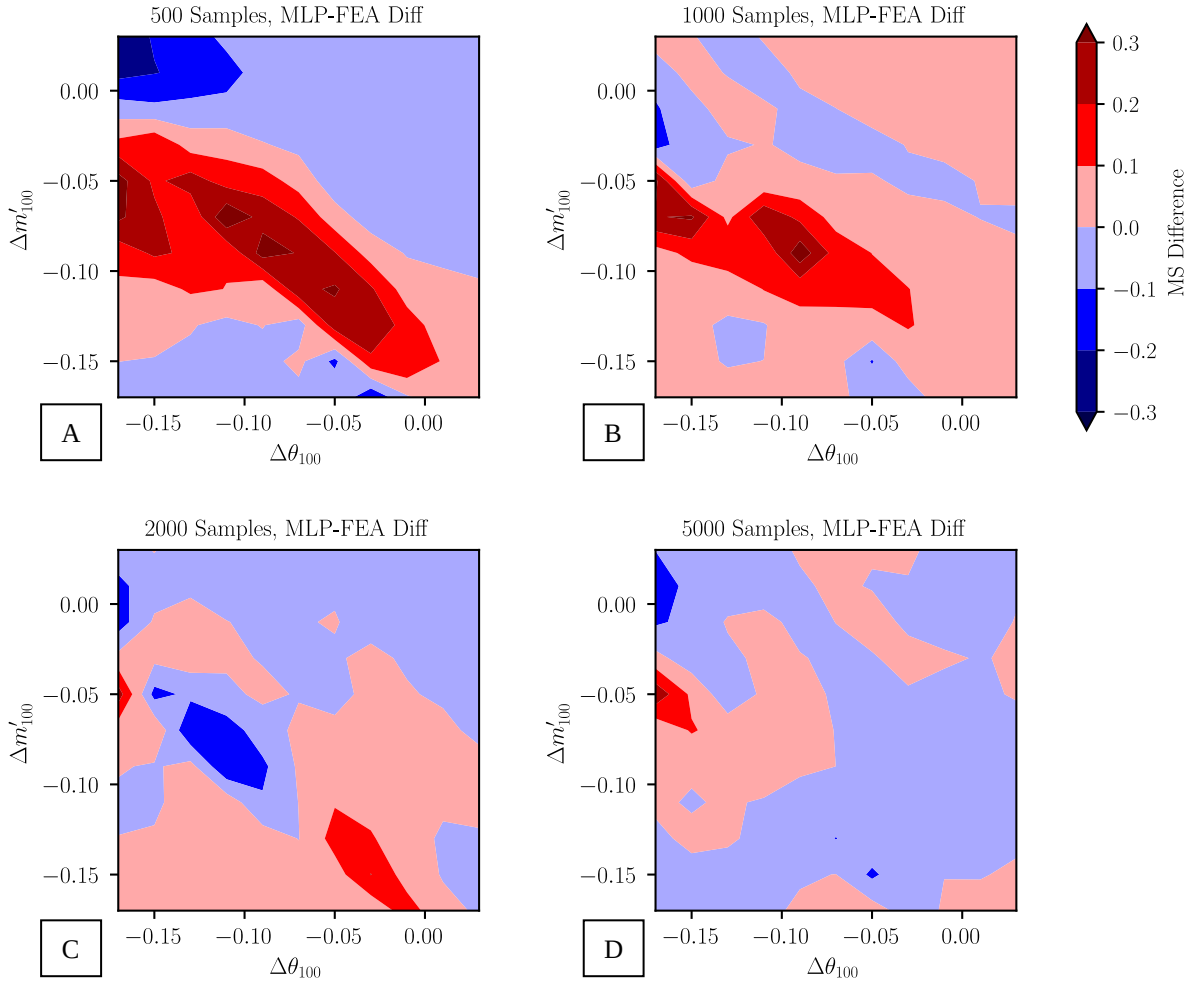

Fig. 12: Difference of MS between MLP model prediction and FEA output. A) 500 sample training set; B) 1000 sample training set; C) 2000 sample training set; D) 5000 sample training set.

Fig. 13 shows the slice of the design space as predicted by the RF models created with the various training sets. Compared to the MLP predicted slice, this result is very clearly different from the FEA output. The result improves as the number of training sets increases, but even the 5000 sample RF model qualitatively performs much poorer compared to the MLP model with the 500 sample training set.

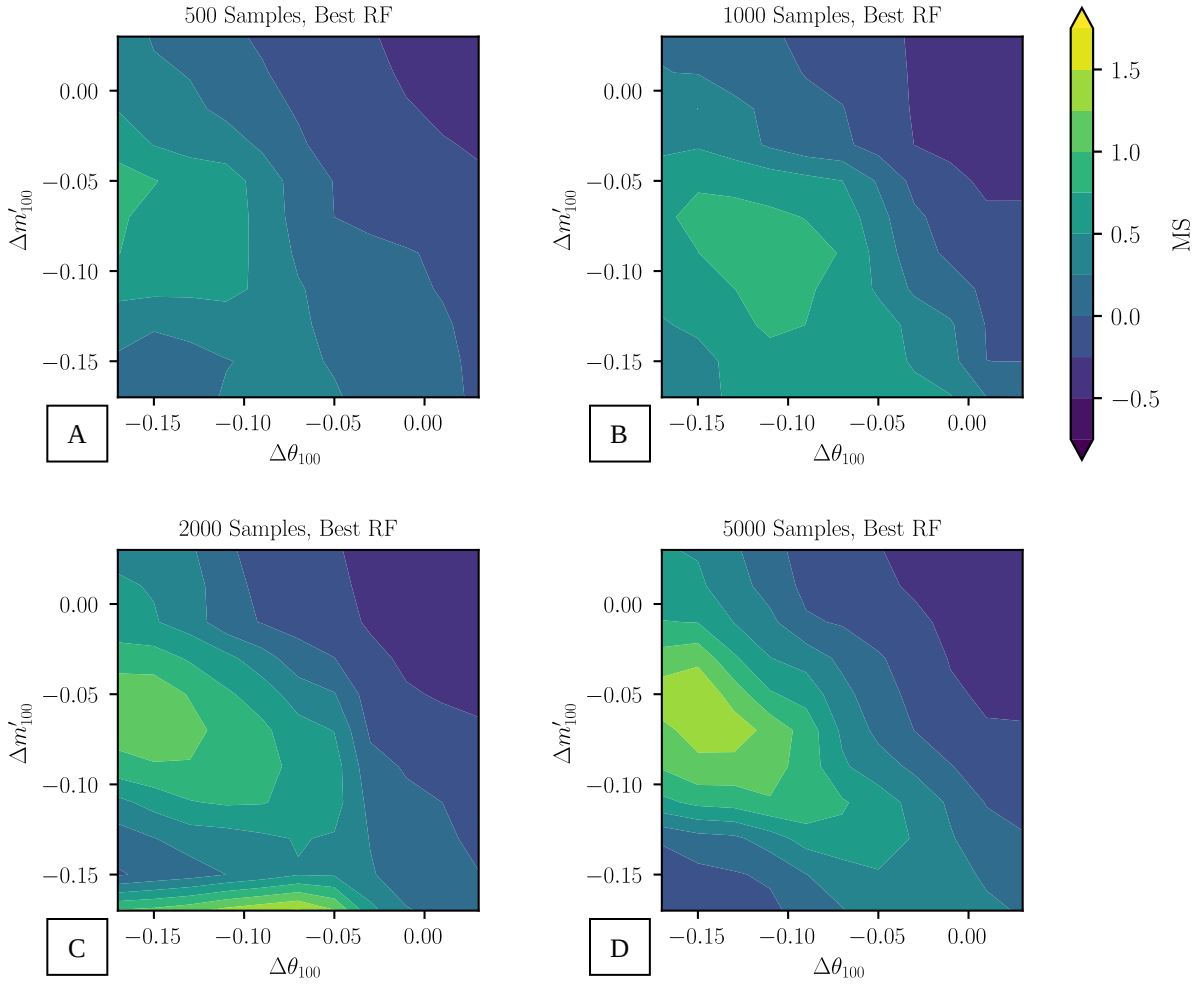

Fig. 13: Design space slices predicted using the best RF models for each training set. A) 500 sample training set; B) 1000 sample training set; C) 2000 sample training set; D) 5000 sample training set.

VI. Model Evaluation for 4 Design Variable

In this section, results for the cases with 4 DVs are briefly discussed. The evaluation in this section focuses only on the R^2 value of the various models. Due to the reduction in design variables from the nominal 6 DV of the previous subsection, it is expected that the surrogate models will be able to better generalize the space will a smaller number of samples.

A. MLP Models, R^2 Evaluation

Fig. 14 below shows the same representation of the accuracy of the MLP models as was shown in Fig. 7 for the 6 DV models. Similar behavior is seen for these models as was shown for the 6 DV cases. The increase in the number of layers from 1 to 3 again shows a significant improvement in the accuracy of the MLP surrogate model while the increase from 3 layers to 5 layers has little effect. Also, it can again be observed that an increase in validation set R^2 value generally leads to a higher R^2 value on the test set. This again indicates that one can make a reasonable selection of surrogate model using only the validation set as a metric of model accuracy.

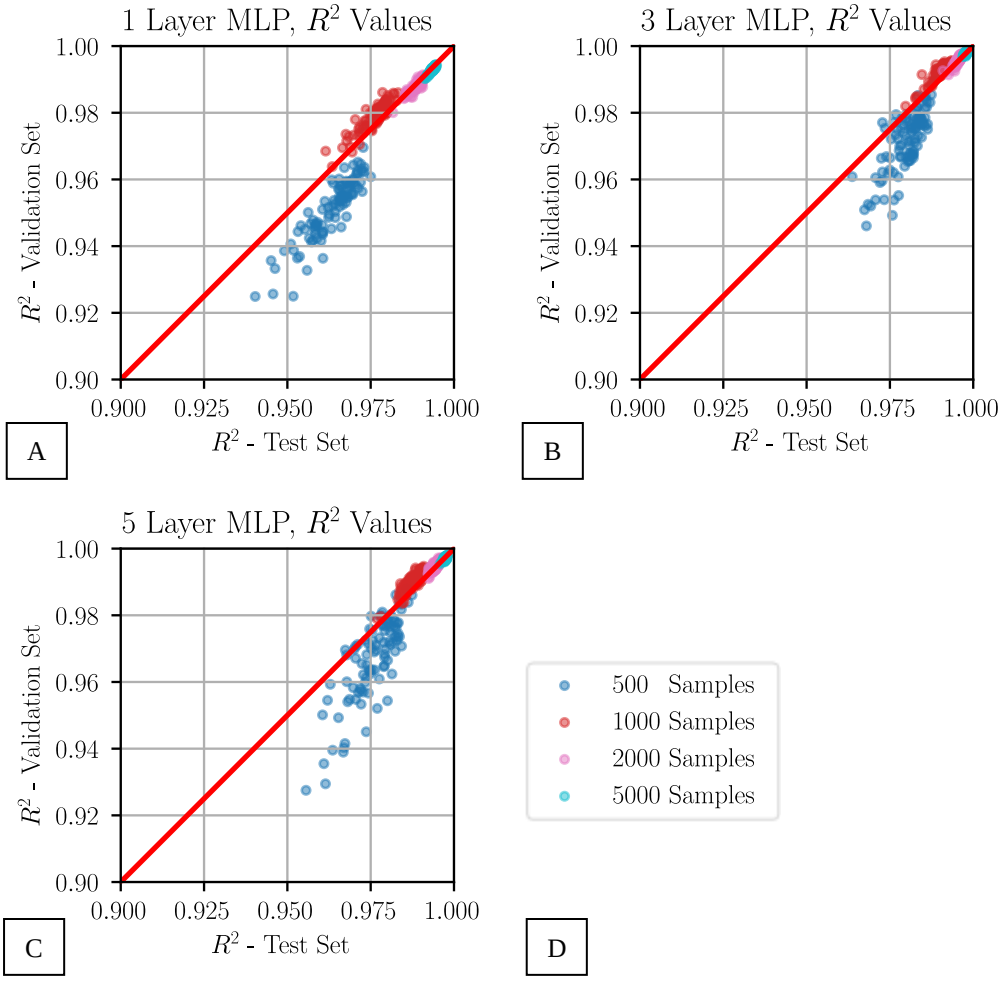

Fig. 14: R^2 values of validation vs test set for all MLP models with 4 DVs. A) 1 layer models; B) 3 layer models; C) 5 layer models; D) Legend showing coloring based on the number of training samples.

Fig. 15, which focuses only on the 3 layer models, provides a clearer evaluation on how a change in the number of DVs impacts model accuracy. In Fig. 15A, which presents models trained with 500 samples, those using 4 DVs consistently achieve significantly higher R^2 values compared to the 6 DV models. With fewer design variables, the model can reach higher accuracy due to a simpler input space and reduced interactions among variables. When training with 500 samples, one of the models with 4 DVs reaches an R^2 value as high as 0.988 on both the test and validation sets while the 6 DV models only ever reach as high as 0.945 on the test set and 0.973 on the validation set. This demonstrates that the 4DV model is able to generalize the space with significantly less training data.

Fig. 15B, when the number of samples is increased to 5000, both 6 DV and 4 DV models perform extremely well with R^2 values typically above 0.99 on both the test and validation set evaluation. Although the 4 DV models continue to perform slightly better than the 6 DV models, the difference is less significant since R^2 values cannot exceed 1. Consequently, diminishing returns become evident, as the marginal improvement in model accuracy no longer justifies the increased cost of generating the FEA data.

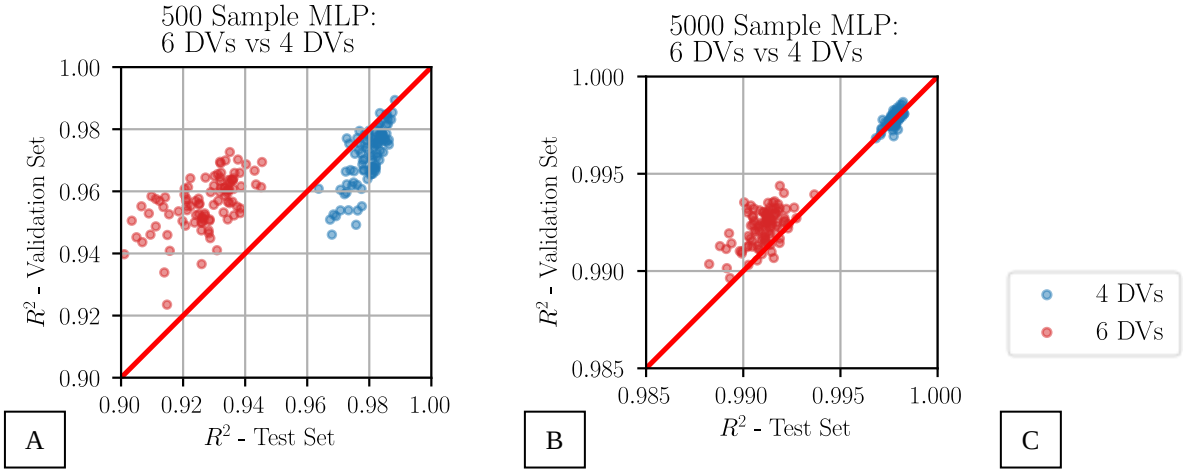

Fig. 15:Comparison of accuracy of 3 layer MLP models for 6 DVs and 4 DVs using R^2 value for validation and test sets. A) Models trained using 500 samples; B) Models trained using 5000 samples; C) Legend to distinguish 4DV models from 6 DV models.

B. RF Model, R^2 Evaluation

Fig. 16 illustrates the R^2 performance of the RF models with 4 DVs. These results are similar to those seen in Fig. 9, showing noticeably poorer performance compared to the MLP models. Like the 6 DV models, even the MLP models trained with only 500 samples demonstrate higher predictive accuracy than the RF models trained with 5000 samples. Like the MLP models, and consistent with expectations for the lower-dimensional space, the 4 DV RF models exhibit better accuracy than their 6 DV counterparts.

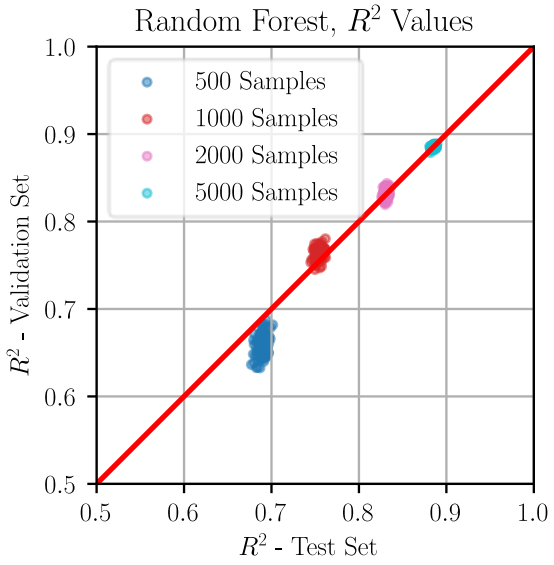

Fig. 16: R^2 values of validation vs test set for all RF models with 4 DVs.

VII. Model Evaluation for 8 Design Variable

In this section, results for the cases with 8 DVs are briefly discussed. The evaluation is this section focuses only on the R^2 value of the various models. Due to the increase in design variables from the nominal 6 DV of the previous subsection, it is expected that the surrogate models will require a larger number of samples to generalize the space. For this design space, the MLP models continue to outperform the RF models. The best RF model trained with 5000 samples performs more poorly on the test set compared to any of the MLP models trained with only 500 samples.

A. MLP Models, R^2 Evaluation

Fig. 17 illustrates the effect of varying the number of hidden layers in the MLP models on accuracy, with the 8 DV configuration exhibiting similar trends to those observed previously. Specifically, increasing the number of layers from 1 to 3 yields a marked improvement in accuracy, while further increasing the layers from 3 to 5 does not result in any significant gain. The comparison of the validation set R^2 value and the test set R^2 value again shows that better performance on the validation set generally indicates better performance on the test set. While it is possible that further increase in DVs may eventually require an increase in the number of layers, the results in this paper have demonstrated that the 3 layer models are sufficiently complex to effectively capture the variable interactions.

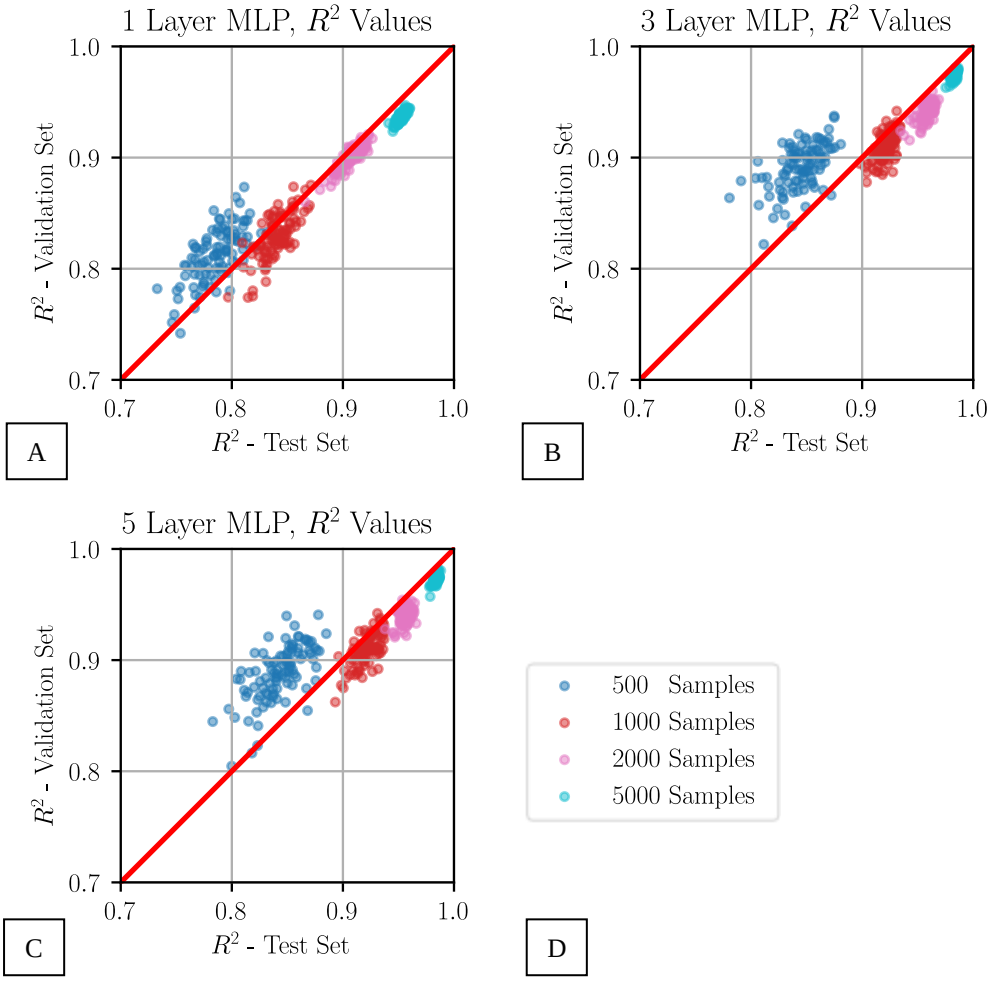

Fig. 17: R^2 values of validation vs test set for all MLP models with 8 DVs. A) 1 layer models; B) 3 layer models; C) 5 layer models; D) Legend showing coloring based on the number of training samples.

Similar to Fig. 15, Fig. 18 illustrates the change in accuracy when moving from the nominal 6 DVs to 8 DVs for the 3 layer models. The results, as expected, show that the 8 DV models achieve lower accuracy compared to the 6 DV models. This is particularly evident in the models trained with 500 samples, a pattern that was also observed in the comparison between the 4 and 6 DV models in Fig. 15. Notably, even with 5000 samples, the 8 DV models still

exhibit a noticeable drop in accuracy compared to their 6 DV counterparts, suggesting that the increased dimensionality may not be fully captured even with the largest training set. Still, the models trained with 5000 samples perform reasonably well with R^2 values as high as 0.985.

This highlights an important consideration in engineering applications: although increasing lean and sweep control points can potentially lead to improved designs, it may also elevate the complexity of the design space to a point where identifying the optimal configuration becomes challenging. Therefore, it is crucial to exercise judgment in maintaining a manageable level of complexity to ensure effective optimization.

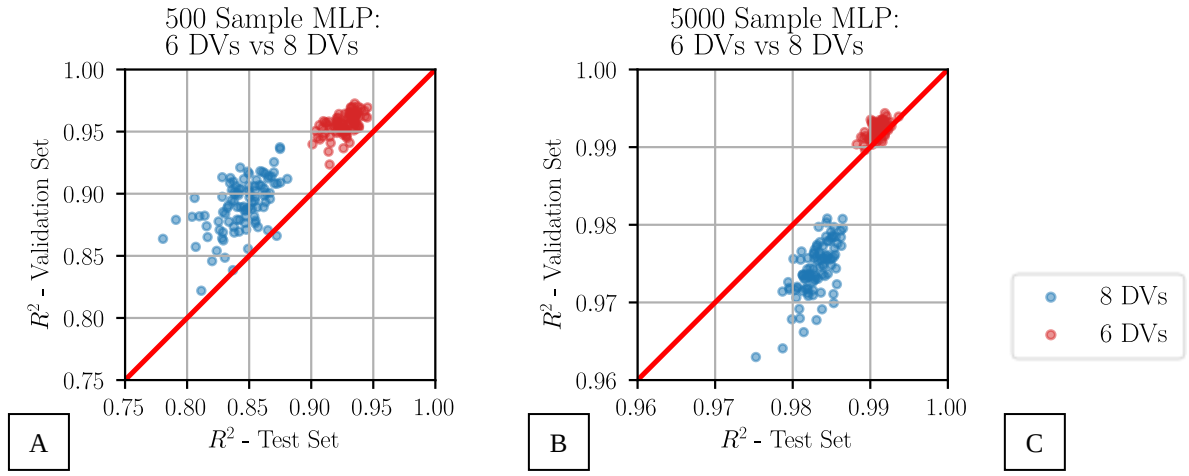

Fig. 18: Comparison of accuracy of 3 layer MLP models for 6 DVs and 8 DVs using R^2 value for validation and test sets. A) Models trained using 500 samples; B) Models trained using 5000 samples; C) Legend to distinguish 8DV models from 6 DV models.

B. RF Models, R^2 Evaluation

Fig. 19 shows the accuracy of the RF models with 8 DVs. These results are similar to those seen in Fig. 8 with noticeably poorer performance compared to the MLP models. As with both the 4 and 6 DV models, even the MLP models trained with only 500 samples demonstrate higher predictive accuracy than the RF models trained with 5000 samples. Consistent with expectations for a higher-dimensional space, the 8 DV RF models exhibit poorer accuracy than their 4 or 6 DV counterparts.

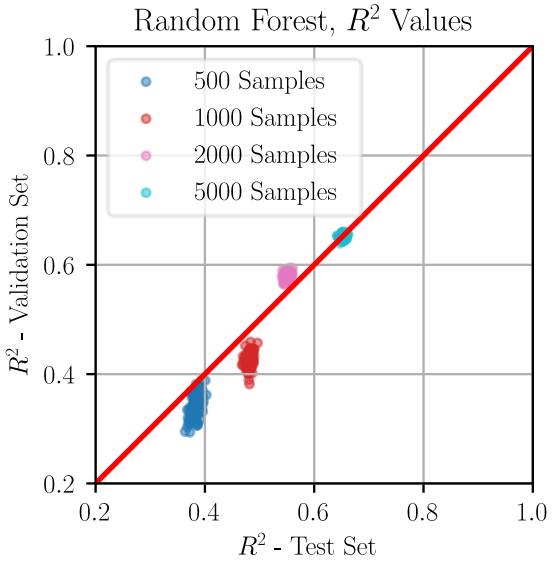

Fig. 19: R^2 values of validation vs test set for all RF models with 8 DVs.

VIII. Optimization Performance

A. Optimization Methodology

Each of the design spaces (4, 6 and 8 DVs) has four sample sizes (500, 1000, 2000 and 5000) that were studied in the previous sections. For each model type (MLP and RF) and combination of design space and sample size, the best performing model in terms of performance on validation set is chosen. That makes a total of 12 MLP and 12 RF models that are chosen for further evaluation through optimization.

To find the optimal point within each MLP and RF model, the differential evolution method for global optimization [15] is used. The implementation of this method that is freely available through SciPy [16] is used in this section. The only inputs that are required are the bounds of the design space and a python function that is used to define the objective function. The differential evolution method does not compute gradients and is similar to genetic algorithm methods. Like genetic algorithms, the differential evolution method requires a larger number of function evaluations than a gradient based method. A typical optimization run performed for this work requires 3000 to 6000 function evaluation. This is not an issue when used on the MLP or RF models since a function evaluation is on the order of milliseconds.

Since the differential evolution method is stochastic in nature, it does not always return the same result. Therefore, the optimization is run 100 separate times with a different random seed for each optimization run. Most of these optimizations return the same result, however, some converge on a local optimum. After the 100 optimization runs are complete, the best point as predicted by the MLP or RF model is chosen for evaluation by FEA simulation.

B. Optimization Results for 6DVs

Table 5 shows the result of the optimization on the MLP and RF surrogate models described in the previous subsection. The results are largely consistent with the findings in the previous section, where the accuracy of the neural network is assessed using the test set. That is, the MLP model generally outperforms the RF model. There is, however, one particular anomaly for the models with 1000 samples. For that sample set, the MLP model predicts a very high MS of 4.571. However, the FEA result obtained from that design had an MS of only 2.133, which is the worst of any of the designs created from optimizing the MLP models.

The anomaly observed in the MLP model with 1000 training samples, where the predicted MS significantly exceeded the actual FEA result, can be attributed to the surrogate model extrapolating near the design-space boundary.

This is evident in Table 6, which shows that $\Delta\theta_{50}$ and $\Delta m'_{100}$ were, respectively, at the boundary and near the boundary. Recall that all DVs have a range of (-0.17,0.03). The steep drop of the MS likely is attributed to the model's prediction of the optimal value for $\Delta m'_{100}$, as that value is notably greater than in the other designs produced by optimizing on the surrogate models.

Put differently, at the edge of the design space, the network lacks support from data points surrounding it on all sides, since no data exist outside the defined bounds. With only 1000 samples, the neural network did not adequately resolve this boundary region, and this resulted in large prediction errors when the network erroneously predicted an increase in MS near the boundary.

While the RF models generally did not perform as well as the MLP models when used in design optimization, none produced as egregiously poor a prediction as the MLP model with 1000 samples. This is likely because RF models inherently limit extrapolation beyond the range of the training data. Instead of making extreme predictions with limited data, they tend to flatten out near the boundaries, which makes them more robust in those regions.

Table 5: MS at the optimal design point as predicted by the ML models for the 6 DV space. Results are shown for both the ML-predicted MS and the corresponding FEA calculation, across both model types and all sample sizes.

Model Type	Sample Size	MS – ML Prediction	MS – FEA Result
MLP	500	2.816	2.606
MLP	1000	4.571	2.133
MLP	2000	3.446	2.830
MLP	5000	3.489	4.475
RF	500	2.419	1.744
RF	1000	2.599	3.362
RF	2000	2.548	2.193
RF	5000	2.903	2.789

Table 6: Design parameters associated with predicted optimal design of various models listed in Table 5.

Model Type	Sample Size	$\Delta\theta_{25}$	$\Delta\theta_{50}$	$\Delta\theta_{100}$	$\Delta m'_{25}$	$\Delta m'_{50}$	$\Delta m'_{100}$
MLP	500	-0.0962	-0.1319	-0.1700	-0.0342	-0.1081	-0.0925
MLP	1000	-0.1277	-0.1700	-0.1253	-0.0131	-0.1168	-0.1645
MLP	2000	-0.1057	-0.1588	-0.1349	-0.0452	-0.1191	-0.1263
MLP	5000	-0.1198	-0.1639	-0.1383	0.0064	-0.0868	-0.1152
RF	500	-0.1010	-0.1227	-0.1585	0.0265	-0.0970	-0.0907
RF	1000	-0.1045	-0.1611	-0.1088	-0.0314	-0.1051	-0.1444
RF	2000	-0.1057	-0.1596	-0.1398	-0.0460	-0.1393	-0.1305
RF	5000	-0.1262	-0.1696	-0.1595	0.0061	-0.1091	-0.1191

C. Optimization Results for 4 DVs

The optimization results from the MLP and RF models for the 4 DV space, presented in Table 7, differ notably from those observed in the 6 DV case. Most significantly, the prediction accuracy for the MLP models is higher in the 4 DV space, with the ML-predicted and FEA-calculated MS values in much closer agreement across all sample sizes. This is consistent with the findings in Section VI A, which showed very high R^2 values for the MLP models with 4 DVs. For the MLP models, the results with 500, 1000, and 2000 training samples each yield better designs, in terms of FEA-calculated MS, than their counterparts in the higher-dimensional space. This improvement highlights the advantage of working in a lower-dimensional design space: with fewer variables, the underlying relationships are less complex, making it easier for surrogate models to learn the space effectively, even with relatively modest sample sizes. This, in turn, leads to more reliable optimization results.

However, the outcome with 5000 samples illustrates the tradeoff inherent in dimensionality. While a lower-dimensional space is easier to capture with limited data, a higher-dimensional space offers greater flexibility, potentially enabling superior designs because there are more variables available to influence the geometry. As seen in

the 6 DV case results in Table 5, the best design achieved with 5000 samples was superior, though this comes at the expense of the increased modeling cost required to generate such a large dataset. Ultimately, while reducing dimensionality aids surrogate modeling and improves optimization efficiency, it can also limit the potential for discovering the best possible design.

Like the MLP results, the RF results were more consistent across different sample sizes for the 4 DV space. As expected, the MLP models generally outperformed the RF models, and, in this case, there is no anomalous result where an MLP model significantly overpredicts MS at a particular point in the space. One notable difference in the optimal designs can be seen in their associated parameters in Table 8. All MLP models tended to drive to the edge of the design space for $\Delta\theta_{50}$. This behavior is also suggestive of extrapolation, similar to what was observed in one of the 6 DV results. However, here, driving toward the boundary actually results in better FEA performance. Whether this is simply fortunate or indicative of improved surrogate accuracy at the boundary due to the lower dimensionality is not entirely clear. This tendency was less pronounced for the RF models, although the 2000 and 5000 sample RF results also trended toward the edge of the space. This likely indicates that, in this case, the edge of the design space genuinely corresponds to an optimal region, and that the MLP models were able to identify this trend effectively, even with fewer training samples.

Table 7: MS at the optimal design point as predicted by the ML models for the 4 DV space. Results are shown for both the ML-predicted MS and the corresponding FEA calculation, across both model types and all sample sizes.

Model Type	Sample Size	MS – ML Prediction	MS – FEA Result
MLP	500	3.267	3.389
MLP	1000	2.969	3.765
MLP	2000	3.348	3.574
MLP	5000	3.690	3.949
RF	500	2.168	3.015
RF	1000	2.318	2.799
RF	2000	2.794	3.057
RF	5000	2.907	3.519

Table 8: Design parameters associated with the optimal design from the various models listed in Table 7

Model Type	Sample Size	$\Delta\theta_{50}$	$\Delta\theta_{100}$	$\Delta m'_{50}$	$\Delta m'_{100}$
MLP	500	-0.1700	-0.1693	-0.0644	-0.0734
MLP	1000	-0.1629	-0.1213	-0.0795	-0.1168
MLP	2000	-0.1700	-0.1445	-0.0705	-0.0939
MLP	5000	-0.1700	-0.1305	-0.0532	-0.1004
RF	500	-0.1482	-0.1525	-0.0706	-0.0704
RF	1000	-0.1399	-0.1087	-0.1146	-0.1222
RF	2000	-0.1665	-0.1491	-0.0823	-0.0926
RF	5000	-0.1597	-0.1252	-0.0484	-0.0992

D. Optimization Results for 8 DVs

Optimization results for the 8 DV space clearly demonstrate the effects of the 'curse of dimensionality,' which is described in Ref. [17]: "One potential issue with surrogate models is the curse of dimensionality, which refers to poor scalability with the number of inputs. The larger the number of inputs, the more model evaluations are needed to construct a surrogate model that is accurate enough." This challenge is evident in Table 9, where only one model, the MLP with 2000 samples, outperforms even the poorest result from the 4 DV space (RF with 1000 samples).

As the design space expands, surrogate modeling becomes increasingly impractical; producing enough samples to build an adequate surrogate is often not feasible. This is particularly true when there is strong variable interaction, as is the case for the DVs chosen for this study. Models with lower sample counts perform especially poorly, with the RF model using 500 samples failing to achieve an MS above zero. The MLP model with 1000 samples also appears to suffer from extrapolation issues. As shown in Table 10, four of the design variables for that model are at the edge

of the design space ($\Delta\theta_{75}$, $\Delta\theta_{100}$, $\Delta m'_{25}$, $\Delta m'_{75}$), and the resulting FEA performance is comparatively poor. Consistent with the trend observed in the 6 DV case, the RF model outperforms the MLP model for this sample set.

In the 8 DV space, there is no apparent benefit from increasing the number of design variables. Models trained with small sample sizes perform very poorly, and even those trained with 5000 samples yield significantly worse MS results compared to the 4 DV and 6 DV cases.

Table 9: MS at the optimal design point as predicted by the ML models for the 8 DV space. Results are shown for both the ML-predicted MS and the corresponding FEA calculation, across both model types and all sample sizes.

Model Type	Sample Size	MS – ML Prediction	MS – FEA Result
MLP	500	2.091	1.616
MLP	1000	1.797	1.186
MLP	2000	2.333	2.996
MLP	5000	2.644	2.377
RF	500	1.208	-0.019
RF	1000	1.273	1.912
RF	2000	1.462	1.065
RF	5000	1.732	2.181

Table 10: Design parameters associated with the optimal design from the various models listed in Table 9.

Model Type	Sample Size	$\Delta\theta_{25}$	$\Delta\theta_{50}$	$\Delta\theta_{75}$	$\Delta\theta_{100}$	$\Delta m'_{25}$	$\Delta m'_{50}$	$\Delta m'_{75}$	$\Delta m'_{100}$
MLP	500	-0.0181	-0.1700	-0.1700	-0.1700	-0.0432	0.0288	-0.0327	-0.0456
MLP	1000	-0.0797	-0.0613	-0.1700	-0.1700	0.0300	-0.0864	0.0300	-0.0283
MLP	2000	-0.0966	-0.1637	-0.0883	-0.1653	0.0300	-0.0651	-0.1426	-0.0803
MLP	5000	-0.0970	-0.0494	-0.1474	-0.1700	0.0230	-0.1664	-0.0301	-0.0539
RF	500	-0.0106	-0.1695	-0.1588	-0.1273	-0.0327	-0.0014	-0.1668	-0.0939
RF	1000	-0.0719	-0.0708	-0.1637	-0.1684	0.0032	-0.0982	0.0208	-0.0334
RF	2000	-0.0108	-0.0537	-0.1272	-0.0864	-0.1698	-0.1363	-0.0537	-0.1390
RF	5000	-0.0202	-0.0840	-0.1464	-0.1250	-0.0376	-0.0754	-0.0456	-0.0721

IX. Conclusion

MLP and RF models were both evaluated for their performance as surrogate models. The MLP models consistently outperformed the RF models in terms of overall accuracy, with considerably higher R^2 values on both the test and validation sets. The analysis revealed a surprisingly wide range of accuracy among MLP models with the same architecture (number of layers), highlighting the advantage of generating multiple models and selecting the best one. An expected and useful observation that should aid in model selection is that, for the MLP models, higher R^2 values on the test set typically corresponded to higher R^2 values on the validation set. Therefore, using the validation set to select the best model should generally yield a better surrogate than simply training a single model. Since these models train very quickly, this approach adds little computational overhead compared to the time required to generate the training data. Regarding architecture selection, increasing from one to three layers led to significant improvements in MLP accuracy, while moving from three to five layers provided little additional benefit, with both architectures producing similar R^2 values. While this observation may not hold for all design spaces, future users of this method can always rely on validation set accuracy as a practical means for model architecture selection.

Despite the overall accuracy advantages, the MLP models are not without drawbacks. After model assessment using test and validation sets, the best models identified by validation performance were selected as surrogates for design optimization. While the MLP models generally outperformed the RF models as surrogates used for design optimization, consistent with the findings of the accuracy analysis, some anomalies were observed. Specifically, two of the MLP models exhibited extrapolation issues, identifying optima at or near the limits of the design space for two or more variables, with FEA results that did not match the surrogate model predictions. While the RF models did not encounter this particular issue, their overall performance in the optimization process remained lower, reflecting their substantially poorer accuracy compared to the MLP models.

Another challenge with the use of these surrogate models, not previously discussed, is the significant amount of data required to achieve acceptable accuracy. While the MLP models trained on 500 samples generally performed well in the 4 DV space, their performance declined sharply in the 6 and 8 DV spaces, with accuracy degrading further as dimensionality increased. This highlights the impact of the 'curse of dimensionality,' a fundamental challenge for all types of surrogate models. Additionally, in an effort to represent the entire space accurately, the current approach uses substantial computational resources on regions of the design space that ultimately yield very poor performance.

The surrogate model approach is best suited to applications where a large number of simulations can be run in parallel. The authors have previously encountered such a scenario when using an in-house NASA CFD code, APNASA [18], which allowed for massive parallelization. While individual simulations still required four or more hours to complete, the use of the NASA Advanced Supercomputer made it possible to submit 1200 LHS-generated designs for analysis simultaneously. Similar to the present study, the performance data from these APNASA simulations were then used to train an MLP model. In that case, the long runtime of individual simulations made it impractical to apply even gradient-based optimization methods directly to the CFD analysis. However, the ability to run simulations in parallel enabled the surrogate model's training data to be generated in roughly one day's time, and it would have been even quicker if not for the need to queue for supercomputer resources. This example underscores the significant advantage provided by both access to supercomputing resources and simulation tools that are free of restrictive licensing requirements.

Unfortunately, running simulations massively in parallel is not always feasible, as the most capable simulation tools often carry substantial licensing costs in addition to computer resource limitations. As a result, future research in our group will focus on optimization methods that require far fewer data points while still maintaining a low risk of becoming trapped in local minima. One promising approach is Bayesian optimization, which has already been applied with great success to the same design spaces explored in this work. Bayesian optimization can achieve comparable optimization results with significantly fewer simulations by efficiently balancing exploration and exploitation in the search space. Moving forward, this strategy offers a practical and scalable alternative for design optimization when computational or licensing constraints make large-scale sampling impractical. Bayesian optimization represents a promising future direction for surrogate-based engineering design.

Appendix

MLP Model Training Function

```
def TrainMlp(TrainingFilePath, ValidationFilePath, RandomState=1, HiddenLayers=(100,100,100)):
    Xtrain, Ytrain, Xvalid, Yvalid = ReadTrainingData(TrainingFilePath, ValidationFilePath,
                                                      Drop=['case','Mass'], Objective=['FS'])

    # Normalize the data using StandardScaler
    scalerX = StandardScaler()
    scalerY = StandardScaler()

    # Fit the scaler on the training data and transform both train and test sets
    Xtrain_scaled  = scalerX.fit_transform(Xtrain.to_numpy())
    Xvalid_scaled  = scalerX.transform(Xvalid.to_numpy())

    # Reshape Y for scaler and transform
    Ytrain_scaled = scalerY.fit_transform(Ytrain.to_numpy().reshape(-1, 1)).ravel()
    Ytest_scaled  = scalerY.transform(Yvalid.to_numpy().reshape(-1, 1)).ravel()

    # Initialize the MLPRegressor model without early stopping but with warm_start
    mlp = MLPRegressor(
        hidden_layer_sizes=HiddenLayers,
        max_iter=1,  # We'll control iterations manually
        random_state=RandomState,
        warm_start=True,  # Allows incremental fitting
        solver='adam',
        learning_rate_init=0.001,
        # learning_rate='adaptive'
    )

    MlpObject = MlpClass(mlp, scalerX, scalerY)
    # Parameters for early stopping
    n_iter_no_change = 100  # Number of epochs with no improvement to wait before stopping
    max_epochs = 20000      # Maximum number of epochs
    best_loss = np.inf     # Initialize best loss to infinity
    epochs_no_improve = 0  # Counter for epochs with no improvement
    train_losses = []
    val_losses = []

    for epoch in range(max_epochs):
        # Train the model for one epoch
        MlpObject.mlp.fit(Xtrain_scaled, Ytrain_scaled)

        # Predict on the test set
        y_val_pred = MlpObject.mlp.predict(Xvalid_scaled)

        # Calculate validation loss (Mean Squared Error)
        val_loss = mean_squared_error(Ytest_scaled, y_val_pred)
        val_losses.append(val_loss)

        # Check for improvement
        if val_loss < best_loss - 1e-5:  # Small threshold to account for floating-point errors
            best_loss = val_loss
            epochs_no_improve = 0
            best_model = MlpObject.mlp  # Save the best model
        else:
            epochs_no_improve += 1

        # Print progress every 1 epochs
        if (epoch + 1) % 1 == 0:
            print(f"Epoch {epoch + 1}/{max_epochs}, Validation Loss: {val_loss:.6f}")

        # Early stopping condition
        if epochs_no_improve >= n_iter_no_change:
            print(f"Early stopping on epoch {epoch + 1}")
            break

    # Use the best model for predictions
    MlpObject.mlp = best_model
    return MlpObject
```

RF Model Training Function

```
def TrainRf(TrainingFilePath, ValidationFilePath, RandomState=42, N_Estimators=100):
    Xtrain, Ytrain, Xvalid, Yvalid = ReadTrainingData(TrainingFilePath, ValidationFilePath,
                                                      Drop=['case','Mass'], Objective=['FS'])

    # Normalize the data using StandardScaler
    scalerX = StandardScaler()
    scalerY = StandardScaler()

    # Fit the scaler on the training data and transform both train and test sets
    Xtrain_scaled  = scalerX.fit_transform(Xtrain.to_numpy())
    Xvalid_scaled  = scalerX.transform(Xvalid.to_numpy())

    # Reshape Y for scaler and transform
    Ytrain_scaled = scalerY.fit_transform(Ytrain.to_numpy().reshape(-1, 1)).ravel()
    Ytest_scaled  = scalerY.transform(Yvalid.to_numpy().reshape(-1, 1)).ravel()

    rf = RandomForestRegressor(n_estimators=N_Estimators,
                               random_state=RandomState
                               )
    rf.fit(Xtrain_scaled, Ytrain_scaled)
    RfObject = RfClass(rf, scalerX, scalerY)
    return RfObject
```


Acknowledgments

This research was sponsored by NASA's Advanced Air Transport Technology (AATT) Project of the Advanced Air Vehicles Program under the Aeronautics Research Mission Directorate. The development of T-Blade3 by the University of Cincinnati's Gas Turbine Simulation Laboratory made this work possible, and the guidance from Mark Turner (NASA-GRC) on T-Blade3 is much appreciated. This work would also not have been possible without the freely available libraries in Python. Therefore, a special thanks to all the code developers who have generously donated their time to make sophisticated scientific and mathematical tools freely available. Finally, the comments from Greg Heinlein (NASA-GRC) during the internal review process improved this paper significantly.

References

[1] S. Braembussche and R. V. d. Pierret, "Turbomachinery Blade Design Using a Navier–Stokes Solver and Artificial Neural Network," *Journal of Turbomachinery*, vol. 121, no. 2, pp. 326-332, 1999.

[2] D. Pasquale, G. Persico and S. Rebay, "Optimization of turbomachinery flow surfaces applying a CFD-based throughflow method," *Journal of Turbomachinery*, vol. 136, no. 3, 2014.

[3] G. Persico, P. Rodriguez-Fernandez and A. Romei, "High-fidelity shape optimization of non-conventional turbomachinery by surrogate evolutionary strategies," *Journal of Turbomachinery*, vol. 141, no. 8, 2019.

[4] K. C. Pierson and M. J. Ha, "Usage of ChatGPT for Engineering Design and Analysis Tool Development," in *AIAA SciTech Forum*, Orlando, 2024, https://arc.aiaa.org/doi/10.2514/6.2024-0914.

[5] P. Mandal, J. Holder, M. G. Turner and M. L. Celestina, "Design and Optimization of a Boundary Layer Ingesting Propulsor," in *ASME Turbo Expo*, London, 2020, https://doi.org/10.1115/GT2020-15603.

[6] "T-blade3 Input Files: Case 7-28-8," University of Cincinnati Gas Turbine Simulation Laboratory, [Online]. Available:https://github.com/GTSL-UC/T-Blade3/tree/master/inputs/OpenCSM/Case7-28-8. [Accessed 16 October 2023].

[7] scikit-learn, "scikit-learn: Machine Learning in Python," scikit-learn project, [Online]. Available: https://scikit-learn.org/stable/. [Accessed 2024].

[8] scikit-learn, "MLPRegressor," scikit-learn project, [Online]. Available:https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html. [Accessed 2024].

[9] scikit-learn, "RandomForestRegressor," scikit-learn project, [Online].

Available:https://scikit-learn.org/dev/modules/generated/sklearn.ensemble.RandomForestRegressor.html. [Accessed 2024].

[10] The SciPy Community, "SciPy Latin Hypercube Sampling," 2023. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.qmc.LatinHypercube.html.

[11] G. T. S. L. University of Cincinnati, "T-Blade3 Github," [Online]. Available: https://github.com/GTSL-UC/T-Blade3. [Accessed 2024].

[12] C. D. Lin and B. Tang, "Latin Hypercubes and Space-filling Designs," *arXiv e-prints,* March 2022, https://doi.org/10.48550/arXiv.2203.06334.

[13] K.-T. Fang, C.-X. Ma and P. Winker, "Centered L2-discrepancy of random sampling and Latin hypercube design, and construction of uniform designs," *Mathematics of Computation,* vol. 71, no. 237, pp. 275-297, 10 2002, dl.acm.org/doi/10.1090/S0025-5718-00-01281-3.

[14] scikit-learn, "r2_score," scikit-learn project, 2024. [Online]. Available: https://scikit-learn.org/dev/modules/generated/sklearn.metrics.r2_score.html.

[15] R. Storn and K. Price, "Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces," *Journal of Global Optimization,* vol. 11, pp. 341-359, 1997.

[16] The SciPy Community, "Differential Evolution Optimization," 2025. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.differential_evolution.html.

[17] J. R. R. A. Martins and A. Ning, Engineering Design Optimization, Cambridge University Press, 2021.

[18] "APNASA on NASA Technology Transfer Program," National Aeronautics and Space Administration, [Online]. Available: https://software.nasa.gov/software/LEW-16855-1. [Accessed 2025].

[19] B. Tang, "Orthogonal Array-Based Latin Hypercubes," *Journal of the American Statistical Association,* vol. 88, no. 424, p. 1392, December 1993.

[20] Meta Open Source, "Ax.dev Documentation, Bayesian Optimization," [Online]. Available: https://ax.dev/docs/bayesopt.html. [Accessed May 2024].

[21] S. Haykin, Neural Networks and Learning Machines, Third ed., Upper Saddle River, NJ: Pearson Education, 2008.

[22] scikit-learn, "SVR," scikit-learn project, [Online]. Available: https://scikit-learn.org/1.5/modules/generated/sklearn.svm.SVR.html. [Accessed 2024].